entry:
```
call void @llvm.dbg.value(metadata ptr %a, metadata !21, metadata
... !DIExpression()), !dbg !27
call void @llvm.dbg.value(metadata ptr %b, metadata !22, metadata
... !DIExpression()), !dbg !27
call void @llvm.dbg.value(metadata ptr %c, metadata !23, metadata
... !DIExpression()), !dbg !27
call void @llvm.dbg.value(metadata i32 %n, metadata !24, metadata
... !DIExpression()), !dbg !27
tail call void asm sideeffect "dmb sy\0A\09orr x3,x3,x3\0A", "~{memory}"()
... #8, !dbg !28, !srcloc !29
call void @llvm.dbg.value(metadata i32 0, metadata !25, metadata
... !DIExpression()), !dbg !30
%cmp11 = icmp sgt i32 %n, 0, !dbg !31
br i1 %cmp11, label %for.body.preheader, label %for.cond.cleanup, !dbg !33
```
T / F

for.body.preheader:
```
%wide.trip.count = zext i32 %n to i64, !dbg !31
br label %for.body.init.1
```

for.body.init.1:
```
call void @llvm.dbg.value(metadata i64 0, metadata !25, metadata
... !DIExpression()), !dbg !30
%rem15.init.1 = and i64 0, 1, !dbg !34
%cmp1.not.init.1 = icmp eq i64 %rem15.init.1, 0, !dbg !34
%indvars.iv.next.init.1 = add nuw nsw i64 0, 1, !dbg !37
%rem15.headerCopy.1.init.1 = and i64 %indvars.iv.next.init.1, 1, !dbg !34
%cmp1.not.headerCopy.1.init.1 = icmp eq i64 %rem15.headerCopy.1.init.1, 0,
... !dbg !34
%indvars.iv.next.latchCopy.1.init.1 = add nuw nsw i64
... %indvars.iv.next.init.1, 1, !dbg !37
%rem15.headerCopy.1.2.init.1 = and i64 %indvars.iv.next.latchCopy.1.init.1,
... 1, !dbg !34
%cmp1.not.headerCopy.1.2.init.1 = icmp eq i64 %rem15.headerCopy.1.2.init.1,
... 0, !dbg !34
%indvars.iv.next.latchCopy.1.2.init.1 = add nuw nsw i64
... %indvars.iv.next.latchCopy.1.init.1, 1, !dbg !37
%rem15.headerCopy.1.2.3.init.1 = and i64
... %indvars.iv.next.latchCopy.1.2.init.1, 1, !dbg !34
%cmp1.not.headerCopy.1.2.3.init.1 = icmp eq i64
... %rem15.headerCopy.1.2.3.init.1, 0, !dbg !34
%indvars.iv.next.latchCopy.1.2.3.init.1 = add nuw nsw i64
... %indvars.iv.next.latchCopy.1.2.init.1, 1, !dbg !37
br label %for.body.init.2
```

for.body.init.2:
```
call void @llvm.dbg.value(metadata i64 0, metadata !25, metadata
... !DIExpression()), !dbg !30
%rem15.init.2 = and i64 %indvars.iv.next.latchCopy.1.2.3.init.1, 1, !dbg !34
%cmp1.not.init.2 = icmp eq i64 %rem15.init.2, 0, !dbg !34
%indvars.iv.next.init.2 = add nuw nsw i64
... %indvars.iv.next.latchCopy.1.2.3.init.1, 1, !dbg !37
%rem15.headerCopy.1.init.2 = and i64 %indvars.iv.next.init.2, 1, !dbg !34
%cmp1.not.headerCopy.1.init.2 = icmp eq i64 %rem15.headerCopy.1.init.2, 0,
... !dbg !34
%indvars.iv.next.latchCopy.1.init.2 = add nuw nsw i64
... %indvars.iv.next.init.2, 1, !dbg !37
%rem15.headerCopy.1.2.init.2 = and i64 %indvars.iv.next.latchCopy.1.init.2,
... 1, !dbg !34
%cmp1.not.headerCopy.1.2.init.2 = icmp eq i64 %rem15.headerCopy.1.2.init.2,
... 0, !dbg !34
%indvars.iv.next.latchCopy.1.2.init.2 = add nuw nsw i64
... %indvars.iv.next.latchCopy.1.init.2, 1, !dbg !37
%rem15.headerCopy.1.2.3.init.2 = and i64
... %indvars.iv.next.latchCopy.1.2.init.2, 1, !dbg !34
%cmp1.not.headerCopy.1.2.3.init.2 = icmp eq i64
... %rem15.headerCopy.1.2.3.init.2, 0, !dbg !34
%indvars.iv.next.latchCopy.1.2.3.init.2 = add nuw nsw i64
... %indvars.iv.next.latchCopy.1.2.init.2, 1, !dbg !37
%0 = insertelement <vscale x 4 x i1> undef, i1 %cmp1.not.init.1, i64 0
%1 = insertelement <vscale x 4 x i1> %0, i1 %cmp1.not.headerCopy.1.init.1,
... i64 1
%2 = insertelement <vscale x 4 x i1> %1, i1 %cmp1.not.headerCopy.1.2.init.1,
... i64 2
%3 = insertelement <vscale x 4 x i1> %2, i1
... %cmp1.not.headerCopy.1.2.3.init.1, i64 3
%4 = insertelement <vscale x 4 x i1> undef, i1 %cmp1.not.init.2, i64 0
%5 = insertelement <vscale x 4 x i1> %4, i1 %cmp1.not.headerCopy.1.init.2,
... i64 1
%6 = insertelement <vscale x 4 x i1> %5, i1 %cmp1.not.headerCopy.1.2.init.2,
... i64 2
%7 = insertelement <vscale x 4 x i1> %6, i1
... %cmp1.not.headerCopy.1.2.3.init.2, i64 3
%8 = call <vscale x 4 x i32> @llvm.aarch64.sve.index.nxv4i32(i32 0, i32 1)
%9 = call <vscale x 4 x i32> @llvm.aarch64.sve.index.nxv4i32(i32 4, i32 1)
br label %for.body
```

for.body:
```
%24 = phi i64 [ %indvars.iv.next.latchCopy.1.2.3, %new.latch ], [ 8,
... %for.body.init.2 ]
%25 = phi <vscale x 4 x i32> [ %78, %new.latch ], [ %8, %for.body.init.2 ]
%26 = phi <vscale x 4 x i1> [ %79, %new.latch ], [ %3, %for.body.init.2 ]
%27 = phi <vscale x 4 x i32> [ %80, %new.latch ], [ %9, %for.body.init.2 ]
%28 = phi <vscale x 4 x i1> [ %81, %new.latch ], [ %7, %for.body.init.2 ]
call void @llvm.dbg.value(metadata i64 0, metadata !25, metadata
... !DIExpression()), !dbg !30
%rem15 = and i64 %24, 1, !dbg !34
%cmp1.not = icmp eq i64 %rem15, 0, !dbg !34
%indvars.iv.next = add nuw nsw i64 %24, 1, !dbg !37
%rem15.headerCopy.1 = and i64 %indvars.iv.next, 1, !dbg !34
%cmp1.not.headerCopy.1 = icmp eq i64 %rem15.headerCopy.1, 0, !dbg !34
%indvars.iv.next.latchCopy.1 = add nuw nsw i64 %indvars.iv.next, 1, !dbg !37
%rem15.headerCopy.1.2 = and i64 %indvars.iv.next.latchCopy.1, 1, !dbg !34
%cmp1.not.headerCopy.1.2 = icmp eq i64 %rem15.headerCopy.1.2, 0, !dbg !34
%indvars.iv.next.latchCopy.1.2 = add nuw nsw i64
... %indvars.iv.next.latchCopy.1, 1, !dbg !37
%rem15.headerCopy.1.2.3 = and i64 %indvars.iv.next.latchCopy.1.2, 1, !dbg !34
%cmp1.not.headerCopy.1.2.3 = icmp eq i64 %rem15.headerCopy.1.2.3, 0, !dbg !34
%indvars.iv.next.latchCopy.1.2.3 = add nuw nsw i64
... %indvars.iv.next.latchCopy.1.2, 1, !dbg !37
%29 = icmp uge i64 %indvars.iv.next.latchCopy.1.2.3, %wide.trip.count, !dbg
... !31
br i1 %29, label %epilogueBlock1, label %permute.decision, !dbg !33,
... !llvm.loop !45
```
T / F

epilogueBlock1:
```
%10 = getelementptr inbounds i32, ptr %a, i64 %24, !dbg !38
%11 = getelementptr inbounds i32, ptr %b, i64 %24, !dbg !40
%12 = getelementptr inbounds i32, ptr %c, i64 %24, !dbg !41
%13 = call <vscale x 4 x i32>
... @llvm.aarch64.sve.ld1.gather.sxtw.index.nxv4i32(<vscale x 4 x i1> %26, ptr
... %10, <vscale x 4 x i32> %25)
%14 = call <vscale x 4 x i32>
... @llvm.aarch64.sve.ld1.gather.sxtw.index.nxv4i32(<vscale x 4 x i1> %26, ptr
... %11, <vscale x 4 x i32> %25)
%15 = call <vscale x 4 x i32> @llvm.aarch64.sve.mul.nxv4i32(<vscale x 4 x
... i1> %26, <vscale x 4 x i32> %14, <vscale x 4 x i32> %13)
call void @llvm.aarch64.sve.st1.scatter.sxtw.nxv4i32(<vscale x 4 x i32> %15,
... <vscale x 4 x i1> %26, ptr %12, <vscale x 4 x i32> %25)
%16 = zext <vscale x 4 x i32> %25 to <vscale x 4 x i64>
call void @llvm.aarch64.sve.st1.scatter.sxtw.nxv4i32(<vscale x 4 x i32> %25,
... <vscale x 4 x i1> %26, ptr %12, <vscale x 4 x i32> %25)
br label %epilogueBlock2
```

epilogueBlock2:
```
%17 = getelementptr inbounds i32, ptr %a, i64 %24, !dbg !38
%18 = getelementptr inbounds i32, ptr %b, i64 %24, !dbg !40
%19 = getelementptr inbounds i32, ptr %c, i64 %24, !dbg !41
%20 = call <vscale x 4 x i32>
... @llvm.aarch64.sve.ld1.gather.sxtw.index.nxv4i32(<vscale x 4 x i1> %28, ptr
... %17, <vscale x 4 x i32> %27)
%21 = call <vscale x 4 x i32>
... @llvm.aarch64.sve.ld1.gather.sxtw.index.nxv4i32(<vscale x 4 x i1> %28, ptr
... %18, <vscale x 4 x i32> %27)
%22 = call <vscale x 4 x i32> @llvm.aarch64.sve.mul.nxv4i32(<vscale x 4 x
... i1> %28, <vscale x 4 x i32> %21, <vscale x 4 x i32> %20)
call void @llvm.aarch64.sve.st1.scatter.sxtw.nxv4i32(<vscale x 4 x i32> %22,
... <vscale x 4 x i1> %28, ptr %19, <vscale x 4 x i32> %27)
%23 = zext <vscale x 4 x i32> %27 to <vscale x 4 x i64>
call void @llvm.aarch64.sve.st1.scatter.sxtw.nxv4i32(<vscale x 4 x i32> %27,
... <vscale x 4 x i1> %28, ptr %19, <vscale x 4 x i32> %27)
br label %for.cond.cleanup
```

permute.decision:
```
%30 = call <vscale x 4 x i1> @llvm.aarch64.sve.ptrue.nxv4i1(i32 4)
%31 = call i64 @llvm.aarch64.sve.cntp.nxv4i1(<vscale x 4 x i1> %30, <vscale
... x 4 x i1> %26)
%32 = call i64 @llvm.aarch64.sve.cntp.nxv4i1(<vscale x 4 x i1> %30, <vscale
... x 4 x i1> %28)
%33 = add i64 %31, %32
%34 = icmp uge i64 %33, 4
br i1 %34, label %lane.gather, label %linearized
```
T / F

lane.gather:
```
%35 = call <vscale x 4 x i1> @llvm.aarch64.sve.ptrue.nxv4i1(i32 4)
%36 = call <vscale x 4 x i32> @llvm.aarch64.sve.compact.nxv4i32(<vscale x 4
... x i1> %26, <vscale x 4 x i32> %25)
%37 = call <vscale x 4 x i32> @llvm.aarch64.sve.compact.nxv4i32(<vscale x 4
... x i1> %28, <vscale x 4 x i32> %27)
%38 = xor <vscale x 4 x i1> %26, shufflevector (<vscale x 4 x i1>
... insertelement (<vscale x 4 x i1> poison, i1 true, i32 0), <vscale x 4 x i1>
... poison, <vscale x 4 x i32> zeroinitializer)
%39 = xor <vscale x 4 x i1> %28, shufflevector (<vscale x 4 x i1>
... insertelement (<vscale x 4 x i1> poison, i1 true, i32 0), <vscale x 4 x i1>
... poison, <vscale x 4 x i32> zeroinitializer)
%40 = call <vscale x 4 x i32> @llvm.aarch64.sve.compact.nxv4i32(<vscale x 4
... x i1> %38, <vscale x 4 x i32> %25)
%41 = call <vscale x 4 x i32> @llvm.aarch64.sve.compact.nxv4i32(<vscale x 4
... x i1> %39, <vscale x 4 x i32> %27)
%42 = call i64 @llvm.aarch64.sve.cntp.nxv4i1(<vscale x 4 x i1> %35, <vscale
... x 4 x i1> %26)
%43 = call <vscale x 4 x i1> @llvm.aarch64.sve.whilelt.nxv4i1.i64(i64 0, i64
... %42)
%44 = call <vscale x 4 x i32> @llvm.aarch64.sve.splice.nxv4i32(<vscale x 4 x
... i1> %43, <vscale x 4 x i32> %36, <vscale x 4 x i32> %37)
%45 = call i64 @llvm.aarch64.sve.cntp.nxv4i1(<vscale x 4 x i1> %35, <vscale
... x 4 x i1> %28)
%46 = call <vscale x 4 x i1> @llvm.aarch64.sve.whilelt.nxv4i1.i64(i64 0, i64
... %45)
%47 = call <vscale x 4 x i32> @llvm.aarch64.sve.splice.nxv4i32(<vscale x 4 x
... i1> %46, <vscale x 4 x i32> %37, <vscale x 4 x i32> %41)
%48 = call i64 @llvm.aarch64.sve.cntp.nxv4i1(<vscale x 4 x i1> %35, <vscale
... x 4 x i1> %38)
%49 = call <vscale x 4 x i1> @llvm.aarch64.sve.whilelt.nxv4i1.i64(i64 0, i64
... %48)
%50 = call <vscale x 4 x i32> @llvm.aarch64.sve.sel.nxv4i32(<vscale x 4 x
... i1> %49, <vscale x 4 x i32> %40, <vscale x 4 x i32> %47)
%51 = xor <vscale x 4 x i1> %49, shufflevector (<vscale x 4 x i1>
... insertelement (<vscale x 4 x i1> poison, i1 true, i32 0), <vscale x 4 x i1>
... poison, <vscale x 4 x i32> zeroinitializer)
%52 = call i64 @llvm.aarch64.sve.cntp.nxv4i1(<vscale x 4 x i1> %35, <vscale
... x 4 x i1> %49)
%53 = sub i64 %45, %48
%54 = add i64 %53, %52
%55 = call <vscale x 4 x i1> @llvm.aarch64.sve.whilelt.nxv4i1.i64(i64 0, i64
... %54)
%56 = xor <vscale x 4 x i1> %55, shufflevector (<vscale x 4 x i1>
... insertelement (<vscale x 4 x i1> poison, i1 true, i32 0), <vscale x 4 x i1>
... poison, <vscale x 4 x i32> zeroinitializer)
%57 = and <vscale x 4 x i1> %51, %55
%58 = and <vscale x 4 x i1> %51, %56
%59 = and <vscale x 4 x i1> %58, %56
%60 = or <vscale x 4 x i1> %57, %59
br label %if.then
```

if.then:
```
%arrayidx = getelementptr inbounds i32, ptr %a, i64 %24, !dbg !38
%arrayidx3 = getelementptr inbounds i32, ptr %b, i64 %24, !dbg !40
%arrayidx5 = getelementptr inbounds i32, ptr %c, i64 %24, !dbg !41
%67 = call <vscale x 4 x i1> @llvm.aarch64.sve.ptrue.nxv4i1(i32 4), !dbg !49
%68 = call <vscale x 4 x i32>
... @llvm.aarch64.sve.ld1.gather.sxtw.index.nxv4i32(<vscale x 4 x i1> %67, ptr
... %arrayidx, <vscale x 4 x i32> %44), !dbg !49
%69 = call <vscale x 4 x i32>
... @llvm.aarch64.sve.ld1.gather.sxtw.index.nxv4i32(<vscale x 4 x i1> %67, ptr
... %arrayidx3, <vscale x 4 x i32> %44), !dbg !49
%70 = call <vscale x 4 x i32> @llvm.aarch64.sve.mul.nxv4i32(<vscale x 4 x
... i1> %67, <vscale x 4 x i32> %69, <vscale x 4 x i32> %68), !dbg !49
call void @llvm.aarch64.sve.st1.scatter.sxtw.nxv4i32(<vscale x 4 x i32> %70,
... <vscale x 4 x i1> %67, ptr %arrayidx5, <vscale x 4 x i32> %44), !dbg !49
%71 = zext <vscale x 4 x i32> %44 to <vscale x 4 x i64>, !dbg !49
call void @llvm.aarch64.sve.st1.scatter.sxtw.nxv4i32(<vscale x 4 x i32> %44,
... <vscale x 4 x i1> %67, ptr %arrayidx5, <vscale x 4 x i32> %44), !dbg !49
%72 = insertelement <vscale x 4 x i1> undef, i1 %cmp1.not, i1 false, !dbg !49
%73 = insertelement <vscale x 4 x i1> %72, i1 %cmp1.not.headerCopy.1, i1
... true, !dbg !49
%74 = insertelement <vscale x 4 x i1> %73, i1 %cmp1.not.headerCopy.1.2, i1
... false, !dbg !49
%75 = insertelement <vscale x 4 x i1> %74, i1 %cmp1.not.headerCopy.1.2.3, i1
... true, !dbg !49
%76 = trunc i64 %24 to i32, !dbg !49
%77 = call <vscale x 4 x i32> @llvm.aarch64.sve.index.nxv4i32(i32 %76, i32
... 1), !dbg !49
br label %new.latch, !dbg !49
```

linearized:
```
%61 = insertelement <vscale x 4 x i1> undef, i1 %cmp1.not, i1 false
%62 = insertelement <vscale x 4 x i1> %61, i1 %cmp1.not.headerCopy.1, i1 true
%63 = insertelement <vscale x 4 x i1> %62, i1 %cmp1.not.headerCopy.1.2, i1
... false
%64 = insertelement <vscale x 4 x i1> %63, i1 %cmp1.not.headerCopy.1.2.3, i1
... true
%65 = trunc i64 %24 to i32
%66 = call <vscale x 4 x i32> @llvm.aarch64.sve.index.nxv4i32(i32 %65, i32 1)
br label %new.latch
```

new.latch:
```
%78 = phi <vscale x 4 x i32> [ %77, %if.then ], [ %25, %linearized ]
%79 = phi <vscale x 4 x i1> [ %75, %if.then ], [ %26, %linearized ]
%80 = phi <vscale x 4 x i32> [ %66, %linearized ], [ %27, %if.then ]
%81 = phi <vscale x 4 x i1> [ %64, %linearized ], [ %60, %if.then ]
br label %for.body
```

for.cond.cleanup:
```
tail call void asm sideeffect "dmb sy\0A\09orr x4,x4,x4\0A", "~{memory}"()
... #8, !dbg !42, !srcloc !43
ret void, !dbg !44
```

CFG for 'foo' function